Chapter 2

I. Phases of SLC
II. SLC models (Chpt 2)

I. SLC
   1. REQ  What to build
   2. SPEC
   2a. planning
   3. DSN: AD
      DD
      ID (Interface)  how to build
   4. Impl. / INT
   5. TST (validation)  >Build
   6. Maintenance
   7. Retirement

A. REQ. is where the clients & developers discuss
   Where (chpt 9)
   - The clients will outline the product as he / she conceptualize it (what they want)
   - The developer is to determine what client’s needs are and any constraints exist (cost, time, technology)
   - Operational REQ: functions of the software
   - NON-operational REQ → Resources (time, cost hardware, experience…..)

B. SPEC is a phase where the agreement between developer & client is drawn up based on the REQ.  (Chpt 10)
   The agreement exactly describes:
   - the functionality of the product
   - list of any constraints (conditions) that the product must satisfy
   ⇒ functional SPEC ⇒ contract between client & developer
   • Data flow technique
   • Data structure oriented Technique
   • OO
   • formal language (Z, VDL…)

C. Design (How)  (chpt 11)
   AD ⇒ determine the internal structure of the products (components)
   DD ⇒ Selection of algorithms
   ID ⇒ What are the Resources (Data) that are exchanged between Components

D. Implementation / INT
   Translate the AD / DD / ID into a programming language
E. Testing (Validation)

V & V  Verification & validation

within a step / phase  confirm

test product itself  test-requirements

F. Maintenance

- Corrective
- Preventive
- Enhancement (perfective)
- Adaptive

II. Software Lifecycle models:

(i) Waterfall model (1970~ Barry Boehm)

Characteristics:
- Each phase is completed before going to the next phase
- Each phase is verified by a SQA Group
- Each phase produces a document
• Changes must be reflected in each step

Adv:
• Very simple
• Each step produces a documentation
• Each Documentation is verified

Disadv:
• Sequential (production)
• The product is seen “late”  
• Back edges possible

(ii) Rapid Prototyping model
Essentially same as the waterfall model
But, we are building prototype(s) in REQ

Adv:
• We see the earlier version of the product
• REQ is clearer  
• less backedges

Disadv:
• Use prototypes for Implementation
• Could be more expansive
• Still sequential mode

(iii) Incremental Model I (RAD)   ← Development
Rapid    Application

REQ / R Prototype
Verify

SPEC
Verify

AD
Verify

Divide into Builds
• DD
• Imp / Valid

[ divide into builds after AD ]
build

Characteristics:
- deliver the product in stages / Builds
- A typical product has 10-50 Builds

Adv:
- Parallelism is possible
- Stop at any time
- Adoption is easier
  (technology transfer)

Disadv:
- Integration can be a problem
  (integrate build 1 & 2……)

(iv) Incremental model II  <Quick>
Faster:

```
<table>
<thead>
<tr>
<th>REQ / RP</th>
<th>SPEC 1</th>
<th>DSN 1</th>
<th>IMP 1</th>
<th>deliver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify</td>
<td>Build 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Build 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPEC 2</td>
<td>......</td>
<td>......</td>
<td>deliver</td>
</tr>
<tr>
<td></td>
<td></td>
<td>......</td>
<td>......</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Build n</td>
<td>......</td>
<td>......</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPEC n</td>
<td>......</td>
<td>......</td>
<td></td>
</tr>
</tbody>
</table>
```

Adv:
- Faster
  (more parallelism)

Disadv:
- Integration becomes more difficult

(v) Incremental model III  <Quick>
(Synchronize & stabilize model. Microsoft)
Hybrid model

```
<table>
<thead>
<tr>
<th>REQ / RP</th>
<th>SPEC 1</th>
<th>DSN 1</th>
<th>IMP 1</th>
<th>build 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify</td>
<td>DSN 2</td>
<td>IMP 2</td>
<td>build 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSN n</td>
<td>IMP n</td>
<td>build n</td>
<td></td>
</tr>
</tbody>
</table>
```

Not full version
- Synchronize between teams in certain time
  - Internal → Integrate earlier
- "freeze" – stabilize the product once it is useful (and move on)

Adv:
- Integration becomes less risky
- Good parallelism

Disadv:
- ?

(vi) Evolutionary model <risk Analysis>
(Spiral model) – Barry Boehm

Characteristics:
- Do Risk Analysis before engineering
- Staff, money, technologies, management

Adv:
- evaluate options (alternatives) before we build it
- identify possible risks and solve
- stop at any moment

Disadv:
- project to be big (RA is done)
- Internal project
- Accurate RA measurement is necessary

(vii) Clean-room / Formal model <more reliable>
- Radical approach to S-Development

Characteristics:
- Formal spec. (formal language)
Automatic programming – failure
Convert Spec → Program language
• 4GL
• formal proof

Q: Which model to use?
A: Depends on
- the project
- the company

Ex.
- water-fold model
- new, Pioneering project → spiral model
- if safety is big factor → formal model